



VoIP services with SEMS



Stefan Sayer
stefan.sayer@iptego.com
www.iptego.com
2009-10-02

- Standard media services in 5 minutes
- Scaling VoIP services
- Developing Applications
- Interfacing and Monitoring

● Install SEMS

- debian

```
/etc/apt/sources.list:  
deb http://ftp.iptel.org/pub/sems/debian/ etch free  
$ apt-get update && apt-get install sems
```

- RHEL/CentOS (EPEL5):

```
$ yum install sems sems-gsm sems-mailbox ...
```

needs: make, g++;
optional: libspeex-dev, libspandsp-dev,
libssl-dev, python-dev
PREFIX=/ to install in /

- From source:

```
$ wget ftp://ftp.iptel.org/pub/sems/sems-1.1.1.tar.gz  
$ tar xzvf sems-1.1.1.tar.gz && cd sems-1.1.1  
$ make && sudo make install
```

● announcements

```
/etc/sems/sems.conf:  
load_plugins=sipctrl;announcement;wav;gsm;ilbc;l16;adpcm;speex  
application=announcement
```



Proxy

```
INVITE sip:user_not_found@announcements.mysip.net  
From: ....  
To: ...  
...
```



SEMS

```
/etc/sems/etc/announcement.conf:  
announce_path=/lib/sems/audio/
```

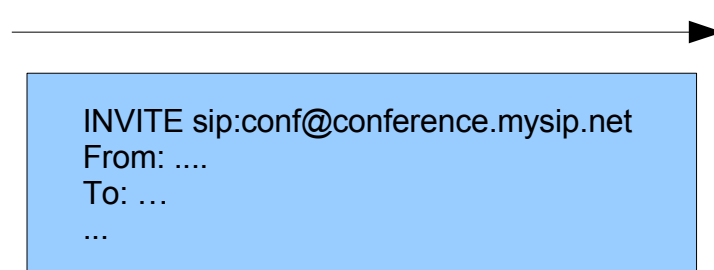
```
/lib/sems/audio/user_not_found.wav  
/lib/sems/audio/announcements.mysip.net  
/lib/sems/audio/announcements.mysip.net/user_not_found.wav  
/lib/sems/audio/announcements.myotherdomain.net  
/lib/sems/audio/announcements.myotherdomain.net/user_not_found.wav
```

● meet-me conference

```
$ cd sems-1.1.1/apps/xmlrpc2di
$ sudo make install; sudo make install-cfg

/etc/sems/sems.conf:
load_plugins=sipctrl;announcement;wav;gsm;ilbc;l16;adpcm;speex;\
  webconference;xmlrpc2di;uac_auth;session_timer
application=$(app_mapping)

/etc/sems/etc/app_mapping.conf:
^sip:.*@announcements.*=>announcement
^sip:.*@conference.*=>webconference
```



You can add a web interface like at
<http://webconference.iptel.org>

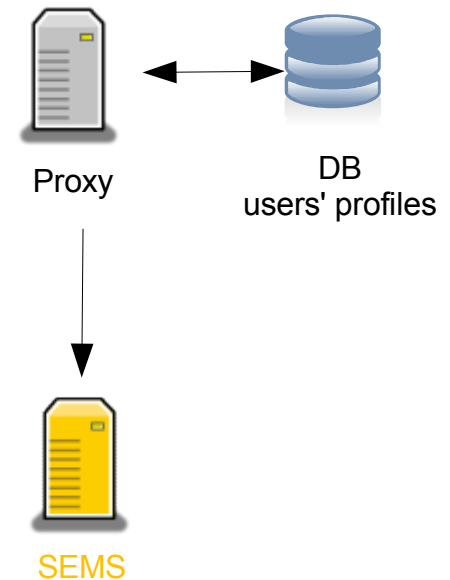
● voicemail, voicebox

/etc/sems/sems.conf:

```
load_plugins=sipctrl;announcement;wav;gsm;ilbc;l16;adpcm;speex;\
webconference;xmlrpc2di;uac_auth;session_timer; \
msg_storage;voicemail;voicebox;annrecorder
```

• **application=\$(apphdr)**

```
if (uri~="^sip:1000.*") {
  append_hf("P-App-Name: voicebox\r\n");
  append_hf("P-App-Param: usr=%@from.uri.user%|;dom=%@from.uri.host%|;
    lng=%$f.lang%|;uid=$f.uid;did=%$f.did%|;\r\n");
  rewritehostport("media.mysip.net:5080");
  route(FORWARD);
}
...
if (lookup_contacts("location")) {
  ...
} else {
  append_hf("P-App-Name: voicemail\r\n");
  append_hf("P-App-Param: mod=%$t.voicemail%|;eml=%$t.email%|;usr=%@ruri.user%|;
    snd=%@from.uri%|;dom=%@ruri.host%|;uid=%$t.uid%|;did=%$t.did%|;\r\n");
  rewritehostport("media.mysip.net:5080");
  route(FORWARD);
}
```



- Performance requirements

- Are you going to have >1.5k channels (g711)?

- Types of applications

- partitioning/sharding possible?

- If not: relay RTP or transfer call behind RR proxy

- (e.g. into conference after room entry)

- Load balancing and failover

- Incoming: monitor and LB on hash(callid)

- Outgoing: monitor and LB on application level

- C++ API
- Python API
- DSM

● C++ API

– Overwrite base application class functions

Name	When called
onStart	session thread is started
onInvite	INVITE received
onCancel	CANCEL received
onEarlySessionStart	session set up in early dialog (183)
onSessionStart	session has been set up, do sth with the stream
onDtmf	Key pressed
onBye	BYE received
onSipRequest	Any SIP request received
onSipReply	Any SIP reply received

– Use functions on any abstraction level

- (“framework” like)

● Example: conference

INVITE

```
void SimpleConferenceDialog::onSessionStart(const AmSipRequest& req)
{
    // set the conference id ('conference room') to user part of ruri
    conf_id = req.user;
    ...
    // get a channel from the status
    channel.reset(AmConferenceStatus::getChannel(conf_id,getLocalTag()));
    // add the channel to our playlist
    play_list.addToPlaylist(new AmPlaylistItem(channel.get(),
                                                channel.get()));
    // set the playlist as input and output
    setInOut(&play_list,&play_list);
    ...
}
```

- Similar to C++ API

- onStart, onSessionStart, onDtmf, onBye, onSipRequest, ...

- Playlist integrated

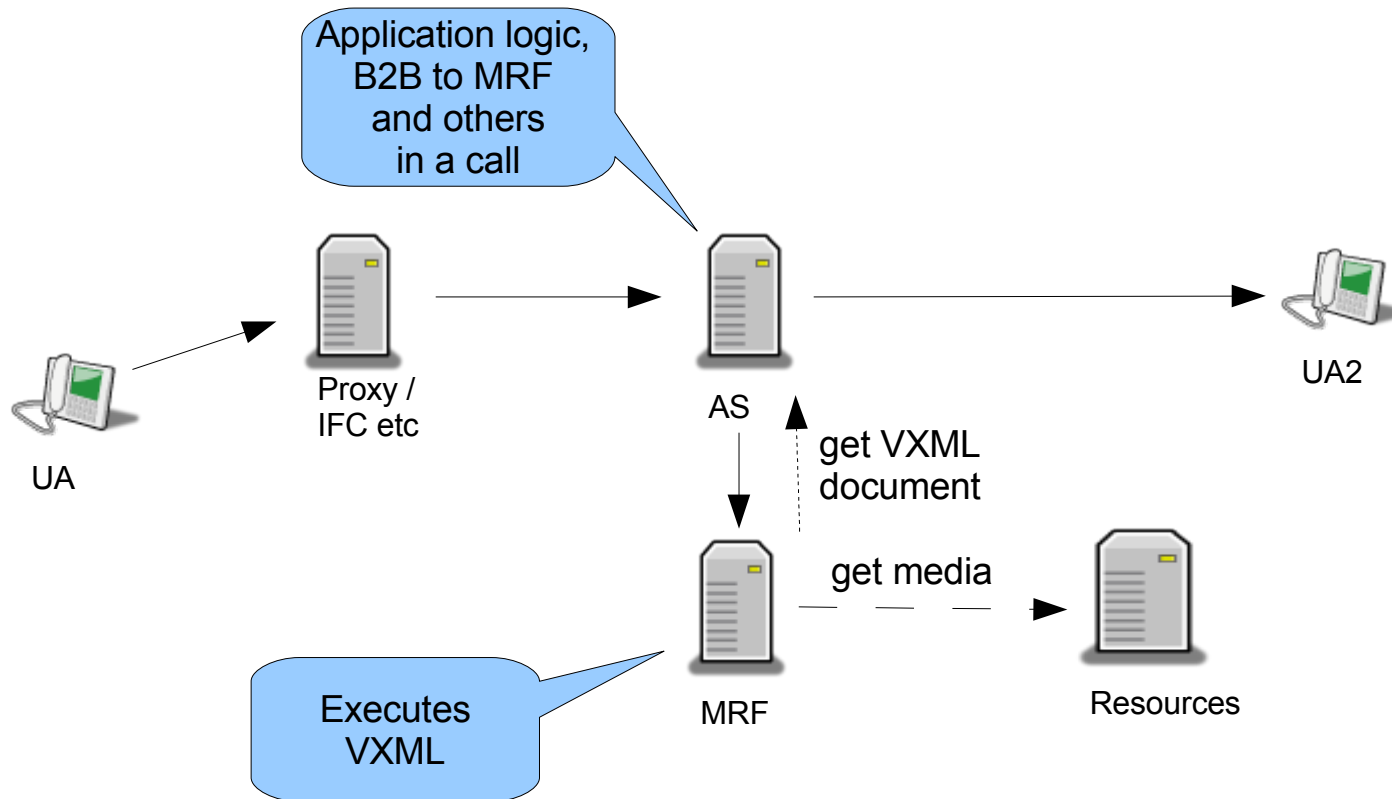
- onEmptyQueue, enqueue(), flush()

- B2BUA

- connectCallee(), terminateOtherLeg(), ...

- Timers

- setTimer(), onTimer,

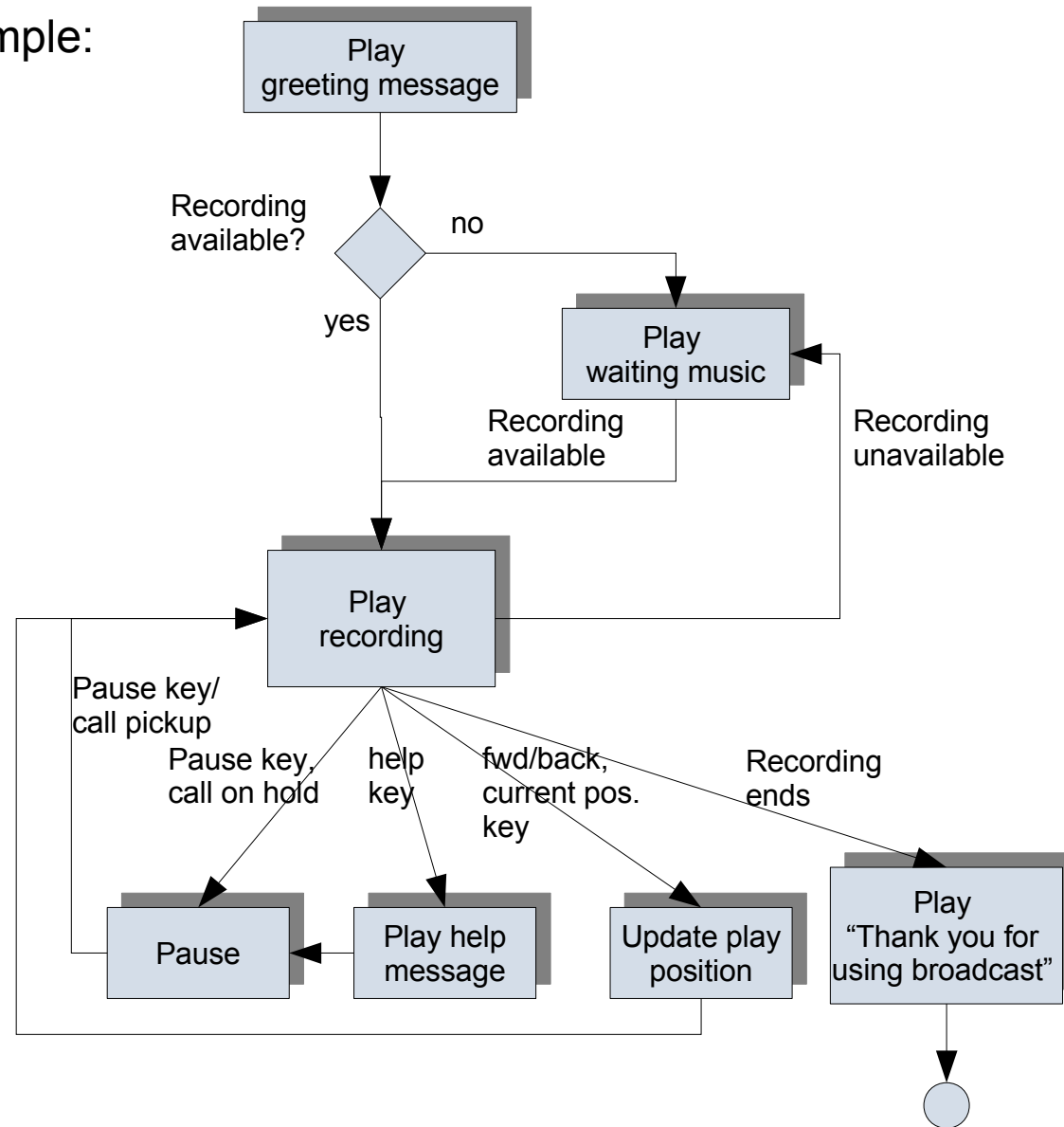


- need to transfer call back and forth
- AS – MRF split makes it complex

Services: how to define

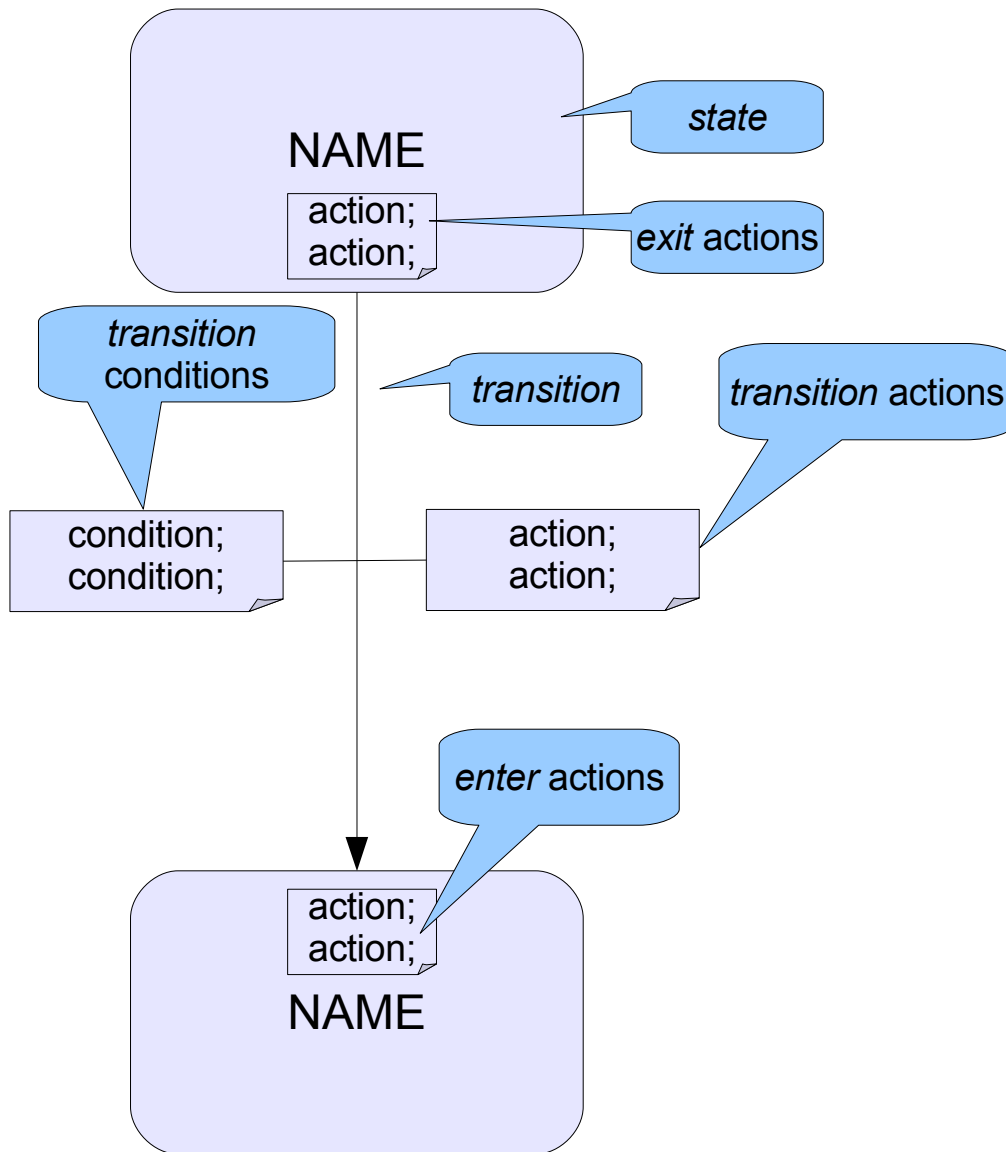
Example:

- State diagrams are the “natural” way to correctly define and document service logic
- Do actions and process events, while transitioning from state to state



see also:

- ECharts for SIP Servlets
- SCXML



DSM state diagram definition language:

```
-- comment
import(mod_name);

[initial] state name
[ enter {
  action;
  action;
  ...
}]
[ exit {
  action;
  action;
  ...
}]
;

transition name s1 - [ { condition; condition; ... } ]
[ / { action; action; ...} ] -> s2;
```

```
-- just a small demo
import(mod_mysql);
import(mod_utils);

initial state START
enter {
    playFile(hello.wav);
    mysql.connect(mysql://user:pwd@domain/db);
    mysql.queryGetResult(select count(id) as new_messages from messages where userid=@user);
};
transition "we have some messages" START - test($new_messages != 0) -> PLAY_MESSAGES;
transition "no messages" - test($new_messages == 0) / playFile(no_messages) -> PLAY_AND_BYE;

state PLAY_AND_BYE;
transition "BYE received" PLAY_AND_BYE - hangup / stop(false) -> end;
transition "files finished" PLAY_AND_BYE - noAudioTest() / stop(true) -> end;

state PLAY_MESSAGES
enter {
    playFile(you_have.wav);
    utils.playcountRight($new_messages);
    playFile(messages.wav);
};
transition "key to listen to the messages" PLAY_MESSAGES - keyPress(1) -> PLAY_MSG;
transition "key to main menu" PLAY_MESSAGES - keyPress(2) -> MENU;
```

- Variables: `$var`
 - > e.g. `set($userid="stefan")`
 - > `test(len($pin)==5)`
- “Selects”: `@select`
 - > e.g. `playFile(@user);`
- Event Parameters: `#param`
 - > e.g. `test(#key==1)`

- Event types

 - >invite, key, timer, noAudio, event (generic), ...

- Hierarchical DSMs

 - >jumpFSM(), callFSM(), returnFSM()

- repost()

 - >mark event as not processed (e.g. transitional states)

- stop()

- Core module: basic actions and conditions
 - > playFile, setPromptSet, playPrompt, recordFile, set, append, log, setTimer, ...
- Modules: more actions and conditions
 - > import(mod_name)
 - > mod.action();

Module	Functionality
mod_sys	System commands (sys.mkdir, ...)
mod_dlg	Dialog related (dlg.bye, ...)
mod_uri	URI processing and operations
mod_conference	Conferencing functions (conference.join,...)
mod_utils	Utilities (utils.getNewId, utils.spell, ...)
mod_monitoring	Monitoring functions
mod_aws	Amazon AWS functions
mod_mysql	MySQL (mysql.query)
mod_py	Python (py(print 'hello world'))

- Implement actions and conditions
- Simplified by macros

```
DEF_ACTION_1P(SCMkDirAction);
...
DEF_CMD("sys.mkdir", SCMkDirAction);
...
EXEC_ACTION_START(SCMkDirAction) {
    string d = resolveVars(arg, sess, sc_sess, event_params);
    DBG("mkdir '%s'\n", d.c_str());
    if (sys_mkdir(d.c_str())) {
        sc_sess->SET_ERRNO(DSM_ERRNO_OK);
    } else {
        sc_sess->SET_ERRNO(DSM_ERRNO_FILE);
    }
} EXEC_ACTION_END;
```

● XMLRPC Events, json-rpc (coming)

```
import(mod_conference)
import(mod_monitoring)

initial state lobby
  enter {
    playFile(wav/welcome.wav)
    monitoring.log(username, @user)
    monitoring.log(position, lobby)
  };

state room;
transition "lobby to room" lobby -
  eventTest(#action==take) / {
  closePlaylist()
  conference.setPlayoutType(adaptive)
  playFile(wav/taken.wav)
  conference.join(#roomname)
  connectMedia()
  monitoring.log(position, #roomname)
} -> room;
```

```
transition "room change" room -
  eventTest(#action==take) / {
  closePlaylist()
  playFile(wav/change.wav)
  conference.join(#roomname)
  connectMedia()
  monitoring.log(position, #roomname)
} -> room;

transition "kickout event" room -
  eventTest(#action==kick) / stop(true) -> end;

transition "bye recvd" (lobby, room) -
  hangup / stop(false) -> end;

state end;
```

```
>>> s.postDSMEvent(call_id, [['action', 'take'], ['roomname', 'wonderland']])
>>> s.listByFilter(['position', 'wonderland'])
```

- DSM state/diagram
- variables map
 - >string variables (map<string, string>)
 - >AmArg (variant) variables (special modules)
- SIP dialog and session
 - >replicate dialog identifiers and RR headers to backup site
 - >for failover send Re-INVITE from backup site (through RR proxy)

- ease development
- make code reusable
- create your service directly from specification
- reduce time to market
 - > adapt services on the fly
 - > speed up implementation/customization – test - product marketing cycle

- Modules can have DI interface (“dynamic invocation”)
 - > Provide functionality between modules
 - > Parameters and return values are variant array (AmArg)
- xmlrpc2di exposes DI APIs as XMLRPC interface (json-rpc coming)
- Useful DI interfaces:
 - > monitoring, webconference, di_dial,
registrar_client, dsm, di_log, call_gen,
cc_acc_xmlrpc

Try it out at:
<https://webconference.iptel.org/control>

- In-memory AVP store
- Identified by string, e.g. related to call (local tag)
- Use with easy macros (e.g. MONITORING_LOG)
- Active flag, optional garbage collector
- STL map as container, AmArg (SEMS' variant type) as data type
- Locked by bucket
- Query by *active*, call local tag, AVP filter, ... also from DI (e.g. XMLRPC)

● Every call

attribute	value
app	application
dir	in/out
ruri	INVITE request-URI
to	INVITE To header
from	INVITE From header

● conference

conf_id	Conference room
---------	-----------------

● DSM

dsm_diag	Current diagram
dsm_state	Current state



Thank you for your
attention.



Stefan Sayer
stefan.sayer@iptego.com
www.iptego.com
2009-10-02